分散型ブラウザにおける通信機能の設計

郭　飛[††]　青沼伴樹[††]　新城靖[††]
佐藤聡[††]　中井央[†††]　板野肯三[††]

1. 背　景

今日、多くの協調アプリケーションが利用されている。協調アプリケーションとは複数のユーザで協調作業をすることができるアプリケーションである。協調アプリケーションを利用することで、複数のユーザがそれぞれのパソコンで同時にドキュメントを編集したり、同時にウェブページを閲覧したりすることができる。

協調アプリケーションの中でも Web アプリケーションとして開発されているものも多く存在する。これらの Web ベースの協調アプリケーションはブラウザを利用することにより利用できる。現在広く利用されている Web アプリケーションとしての協調アプリケーションの多くは中央のサーバを用いて実現される。これにより、潜在的なスケーラビリティが低いこと、セキュリティやプライバシが維持できない問題が発生すること、サービスが終了した時に個人のデータが取り出せなくなるといった問題が発生する。これらの問題を解決するために、我々は、協調アプリケーションの実行環境として分散型ブラウザを提案している[1)]。分散型ブラウザは、マルチユーザのブラウザであり、ユーザの認証、ブラウザ間の通信、ブラウザ上のプロセス管理、分散ストレージなどの機能を提供する。ユーザは協調作業行いたいとき、分散型ブラウザ上で協調アプリケーションを実行する。分散型ブラウザ上で動作する協調アプリケーションはデータを中央のサーバに置く必要がなく、上記の問題を解決することができる。

本論文では分散型ブラウザの通信機能の設計について述べる。

2. 現在の分散型ブラウザの通信機能とその問題点

現在の分散型ブラウザは、通信機能として、Remote Procedure Call (RPC) に基づくものを提供している[1)]。協調アプリケーションは、RPC を用いてクライアント・サーバ・モデルに基づき JavaScript により記述される。通常の RPC とは異なり、分散型ブラウザの RPC は、非同期的なインタフェースになっている。これは、JavaScrip のイベント駆動のモデルに適合させるためである。この RPC は、インスタント・メッセンジャ Skype が提供している AP2AP (Application to Application) API (Application Program Interface) を利用して実装されている。協調アプリケーションは、通信相手を Skype のユーザ名で識別することができる。

この RPC に基づく通信機能を用いて、次のようなアプリケーションの開発を行った。

- 簡単な協調ブラウジング
- 簡単なコメント共有
- ブラウザのスクリーンショットの送受信

これらのアプリケーションの開発を通じて、RPC に基づく通信の有用性は確認された。しかし、現在の分散型ブラウザの通信機能には次のような問題点が残されている。

- RPC だけでは記述しにくいアプリケーションが存在する。たとえば、協調動画視聴のようなアプリケーションでは、サーバからのプッシュ型の通信やグループ通信機能が必要であるが、RPC ではうまく記述することができない。
- Skype の中央のサーバが提供している認証機能に依存している。Skype の認証機能が利用できなくなると、分散型ブラウザの機能も利用できなくなる。

前者の問題を解決するために、本研究では、RPC に加えて WebSocket[2)] に基づく通信を提供する。くわしくは、3 章で述べる。

後者の問題を解決するために、本研究では、複数のインスタント・メッセージング・システム (Instant Messaging System, IMS)、または、ソーシャル・ネットワーク・サービス (Social Network Service, SNS) を利用して分散型ブラウザの通信機能を実現する。

全体の構造は図 1 のようになる。複数の IMS、または、SNS を利用することで、それらのうち 1 つでも利用可能であれば、本分散型ブラウザを利用しつづけることができるようになる。

3. WebSocket に基づく通信方式

WebSocket は W3C と IEFT で標準化が進められている Web クライアントとサーバ間の通信のための API およびプロトコルである。WebSocket では、クライアントからもサーハ からもテ ータの送信することか て き るので、サーバからのデータのプッシュを実現することも容易になる。ただし、Web フ ラウサ 上

† 筑波大学情報学群情報科学類
†† 筑波大学システム情報工学研究科
††† 筑波大学図書館情報メディア研究科

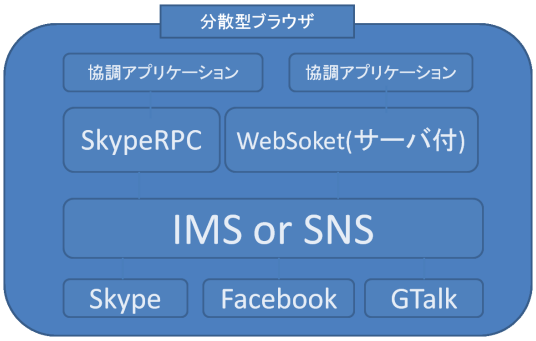

図 1 分散型ブラウザの通信方式

の JavaScript はクライアント側の API しか利用て きない。そのため、ブラウザ間の通信は必す 中央のサーハ を介して行われることに なる。従って、本研究の目標て ある中央のサーハ に依存しないということは達成て きない。そこで本研究は JavaScript 側にサーハ の API を提供する。

WebSocket を分散型フ ラウサ から使うためのクライアント側の API は以下のよ うになる。利用可能なメソッドは、通常の WebSocket API と同じである。

```
1  var s = new WebSocket(server);
2  s.onopen = function() {...};
3  s.onmessage = function (m) {
4     var received msg = m.data;
5     do something(msg);
6  };
7  s.onclose = function() {...};
8  s.send(msg);
```

WebSocket を分散型フ ラウサ から使うためのサーバ側の API は以下のようになる。利用可能なメソッドは、Node.js ☆ を参考にして設計した。

```
1  var server = ss.createServer(function (stream){
2     stream.on('connect', function (){
3        var c = stream.peername;
4        var msg = do_something(c);
5        stream.write(msg);
6     });
7     stream.on('data', function(data){
8        var msg = do_something(data);
9        stream.write(msg);
10    });
11    stream.on('end', function(){
12     stream.end();
13    });
14 });
15 server.listen(app_name);
```

一般の Web ブラウザの WebSocket て はクライアントは URL を指定して接続を行う。分散型フ ラウサ の場合、クライアントは、ユーサ 情報を利用し接続先の指定 することができる。また、クライアントもサーバもメッセーシ 送信者を認証する ことができる。分散型ブラウザは、IMS や SNS のユーサ 情報を利用し、認証を 行う。

ここで User1 と User2 の 2 人のユーサ か 協調作業を行うことを考える。User1 はサーハ 、User2 はクライアントを実行するものとする。この場合ます 、User1 はサーハ 実行す

```
1  var server = ss.createServer(...);
2  server.listen("/example");
```

すると、サーバ側の分散型ブラウザは、ユーザ名 User1 で IMS あるいは SNS に ロク インし、User1 のコンタクトリスとを取得し、クライアントからの接続要求 を待つ。

User2 がクライアント側のプログラムを実行する。

```
1  var s = new WebSocket("ws://User1/example");
```

このように、クライアント側では、ユーザ名で接続先を指定することができる。

サーバ側では、クライアント側から接続要求が来ると、分散型は、User1 のコン タクトリストを調べる。クライアントのユーザ名がコンタクトリストにない場合 には、接続を拒否する。コンタクトにあれば、分散型ブラウザは、次のコール バック関数を呼び出す。

```
1  stream.on('connect', function (){
2     if(stream.peername=='User2') do_something()
          ;
3  });
```

サーバ側のアプリケーションは、コールバック関数の中で peername という フィールドを参照することでクライアントのユーザ名を得ることができる。

4. ま と め

この論文では、分散型ブラウザの通信機能の設計について述べた。その特徴は、 複数の IMS や SNS を利用することにより、可用性を高めていること、および、ブラウザ側で WebSocket のサーバ側の API が利用可能になっている点にある。

今後の課題は、設計した通信機能を実現し、評価することである。

参 考 文 献

1) Yasushi Shinjo, Fei Guo, Naoya Kaneko, Takejiro Matsuyama,Tatsuya Taniuchi, Akira Sato:"A Distributed Web Browser asa Platform for Running Collaborative Applications", The 7thInternational Conference on Collaborative Computing:Networking, Applications and Worksharing (CollaborateCom2011) (2011).
2) I. Hickson. The WebSocket API. http://dev.w3.org/html5/websockets/

☆ http://nodejs.org/